Still Alive With Sir Clive!

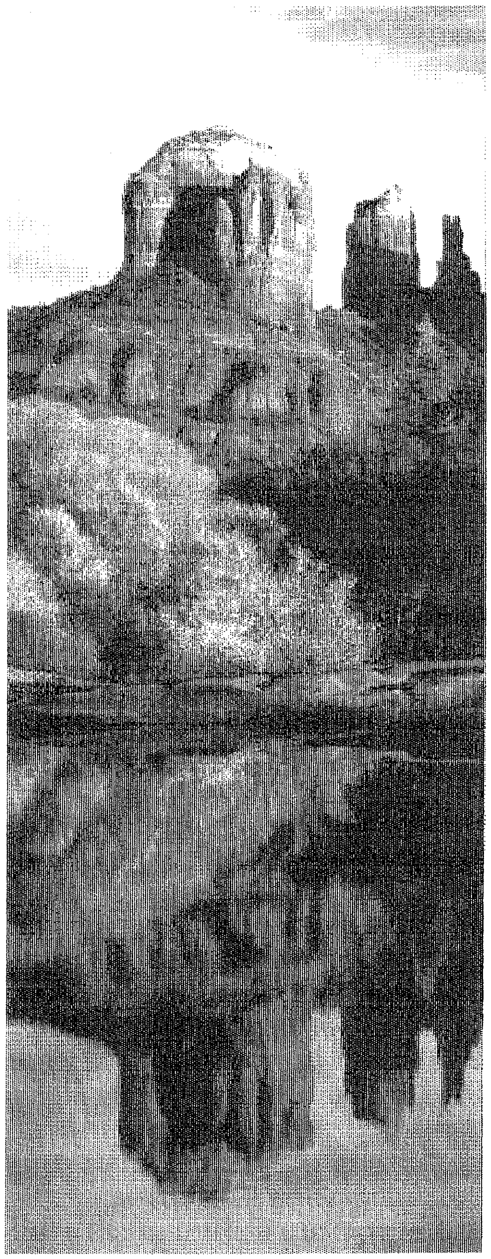
ZXir QLive Alive!

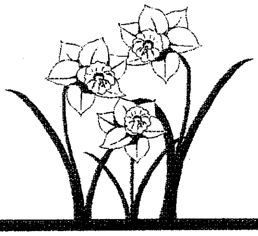
The Timex/Sinclair NorthAmerican User Groups Newsletter

Volume 8 No. 3 Autumn '98

Chairman Donald S. Lambert Auburn, IN

MEMORY MAP





ZXir QLive Alive! ©

Established 1991 The Timex/Sinclair NorthAmerican User Groups Newsletter

T/SNUG Information

We wish to support the following platforms : ZX-80/81, TS-1000, Spectrum, TS-2068, Z88 and QL. If you have any questions about any of these fine Sinclairs, contact the:

Chairman

Chief Motivator
Donald S. Lambert (ISTUG)

Vice-Chairmen

Tape & JLO PD Library
D. G. Smith
415 Stone St.
Johnstown, PA 15906
814 535-6998

Z88 Library
Dave Bennett (HATSUG)
1275 Timber View Dr.
Mechanicsburg, PA 17055-9146
717 732-4374

QL Hacker's Journal
Timothy Swenson
38725 Lexington St.
Fremont, CA 94536
swensontc@geocities.com

TS-2068
Rod Humphreys (VSUG)
10984 Collins Pl.
Delta, BC V4C 7E6 Canada
604 583-2819

QL PD Library
John Donaldson (CATUG)
835 Foxwood Cir.
Geneva, IL 60134-1631
630 232-6147

AERCO & Z80 Emulator
Keith Watson
41634 Amberly Dr.
Mt. Clemens, MI 48038

BBS --==GATOR==--
Bob Swoger (CATUG)
613 Parkside Cir.
Streamwood, IL 60107-1647
630 837-7957 Work 847 576-8068

Any of the above can also be reached by E-Mail through the **Club BBS 847 632-5558**

ZXir QLive Alive!

Is the newsletter of T/SNUG, the Timex/Sinclair North American User Groups, providing news and software support to the T/S community in a **VOLUME** of four newsletters per year; beginning with the Spring (March) issue.

T/SNUG's main goal is to preserve and encourage the use of Sinclair computers by providing an open forum for the exchange of knowledge, building and maintaining of software libraries. Providing vendors, repair service and members with free ad space.

It is the user groups and individual subscribers, rather than the vendors, that provide the pecuniary support for this newsletter. Vendors and developers receive this newsletter free of charge, though contribution from vendors and user groups is gratefully accepted. Please support our vendors and service providers whenever possible.

If you have a problem or you have solved a problem, please share it with the rest of us. No problem will be considered unimportant.

Editor/Treasurer Publisher

You can keep T/SNUG alive by an annual contribution of $12 for one VOLUME made payable to Abed Kahale. Send check to:-

ABED KAHALE
3343 S FLAT ROCK CT
SIERRA VISTA AZ **85650**-6874
520 378-3424

New

Back copies are available for $1.00 each postpaid.

Trea$ury Note$

As of August 30, 1998, we have a balance of $807.32

Article Contributions

Send in your articles by tape or disk and your inputs to:—

DONALD S LAMBERT
1301 KIBLINGER PL
AUBURN IN 46706-3010
Phone 219 925-1372

By hardcopy, MSDOS or modem (.3-33.6) to:
Abed Kahale
E-mail: AKahale@compuserve.com

GATOR'S Twisted Pair

To better inform the Sinclair Community, four 24-hour a day BBSs are now provided to serve you. You are encouraged to exchange mail and use the files sections of these boards. Bulletins and ads are available to all.

Q-Box BBS 810 254-9878
Utica, Michigan
SCC Sever Jose Moreno
http://members.tripod.com/~helpme/
SOL BBS 520 882-0388
Tucson, Arizona
Club BBS 847 632-5558
Arlington Heights, Illinois

If you know the Internet E-Mail address of a Sinclair user, but do not have access to Internet, simply address your E-Mail to GATOR Sinclair on the 24-hour **Club BBS** and include the name and E-Mail address of the user you wish to reach. Then check the Club BBS from time to time if you expect a reply.

We encourage you to exchange mail and contribute to the UPLOAD section. Call and ***register*** using your first, last name and phone number along with a password you won't forget. ***Write It Down!*** Do not try to do anything else at this time.

When you call-in the next time, you will have Level 5 security and be able to enjoy full user privileges. The BBS has smaller sections called conferences. Select "J" for "Join a Conference". Select "TIMEX" to get into the Sinclair Section. The mail you then read will only be from other T/S users. Use extension .ART for articles, .ADS for ads and .NWS for news when UPLOADing.

For help, contact the SYSOP, Bob Swoger, by leaving a message, mail, E-Mail or phone.
CENG108@email.mot.com

Input/Output

by Abed Kahale

Hello Abed,

I just received the V8#2 issue. I am still amazed at all the info you folks pack into it. I also want to add my email address to the list: (panderson2@home.com). It's been a while since I had any of my Timex stuff out, but I enjoy ZQA tremendously. However, after reading this issue, there certainly seems to be a lot of diehards out there keeping TS computers alive. Sinclairly,

Paul M. Anderson

Abed.

I just read Joan Kealy's letter in the ZQA, and I've got to say that I agree with her. You really are doing a great job for the group. I don't know where [illegible] find the time.

Speaking of time (saving a little) you don't need to reply to this. <G>

P.S. I agree with you also - I enjoyed those programs (the ones that I have seen) that Joan wrote for the 2068. I even tried to get her to write some bagpipe music, but by that time she was no longer involved with the 2068. Too bad. Take care.

Dennis Smith

Dear Abed :

We have been trying to bring 2068 telecommunicating capability "into the third millennium," by updating the underlying Z80 CODE to SEND files to modern computers. This effort is proceeding slowly. Another, unrelated problem causes the "cut off" of connections, while entering and/or editing messages. We have traced this phenomenon to a lack of free RAM on the part of the 2068, when running MaxCom BASIC. So, we have reluctantly introduced memory-saving techniques, using slower instructions. Well, the upshot is, that the computer does not operate too much slower, and we want to emphasize a nasty little problem, which has been mentioned before in passing. In order to save memory, we ordinarily go through the code, replacing every occurrence of the numbers 0, 1, and 3 with NOT PI, SGN PI, and INT PI, respective[illegible] But, suppose the code is IF 0 < x THEN GOTO 5 and the value of X is 7. Then, we desire that the instruction take the branch to line 5, but the program will fall through to the next line, unless we use a couple of parentheses, as in

```
        IF (NOT PI) < X THEN GOTO 5
```

Well, a better solution is to replace such a number 0 with CODE "", as in

```
         IF CODE "" < X THEN GOTO 5
```

Then, we get to use "memory-saving code," without any extra parens!

K e e p O n T i m e x 'n

David E. Lassov: Sysop

SOL BBS @ 520-882-0388 (data)

Dear Abed:

I don't recall seeing that anyone has mentioned the scenic photos of Arizona that appear from time to time in ZQA! but I, for one, appreciate them. They remind me of the times 35 - 40 years ago when I took many of the slides in my own collection from that area.

While looking through my file copy of the listing for the BOOT routine that appeared in the Vol. 7 No. 4 ZQA! I discovered a typo (mine, not yours) that would halt it with an error at line 42. The first variable name should be slc, not sls, so the corrected line would be:

```
        42 slceslc-47-39*(slc>57)
```

My apologies for the frustrations I caused anyone who couldn't get the routine to work.

Many have expressed concern that Sinclair enthusiasts are becoming a passing generation. Not to worry, if a note about Marcel Kilgus in the May-June QL Today is any indication. It states that he began development of his excellent QPC emulator program a few years ago when he was 16.

Near the other end of the age spectrum, an article in the same issue by Darren Branagh describes one of the recent inventions by Sir Clive. It is advertised as the world's smallest FM radio. It is entirely contained in an ear-plug shaped to fit into the ear. Tuning is done by touching a tiny scanning button.

With best regards,

Bob Hartung

.... When I needed an affordable computer to learn programming, the ZX-80 kit was there for $99. No more putting up with TimeShare at work. Abed.

My guess is that millions of folks were introduced to "computers" by Sinclair and Timex. I wonder what our computer world would be like ... if Timex had continued developing and marketing their computer products? Do you think we might have a world full of "programmers? Today ... folks read thick books and/or go to seminars to learn how to use a "given" program on their IBM type computers (so the person has to adapt to the computer's programs). We adapted the programs (& computer) to our individual needs. That, to me, was the BIG difference that Sir Clive Sinclair brought into being. The Editorial in the current Popular Electronics makes reference to the fact that many computers in the govt. and business areas will have to be reprogrammed when the year 2000 emerges ... and, in that, most of programs are in the Fortran or Cobol language ... it will be necessary to "call" retired programmers (as very few of the current programmers know those "obsolete" languages).

If I were one of the "retired" programmers called ... I believe my asking price would be exorbitant ... but in that I'm not a "retired" Fortran or Cobol programmer ... I'll never have the chance to "gouge" the industries (oh! Sadness!). Sincerely,

Fred Henn

More NASA Space Images Now Online

NASA's Observatorium's Gallery expands its collection

See new photos of the solar system, Moon, Earth, and more at http://observe.ivv.nasa.gov/nasa/gallery/image_gallery/image_gallery.html

NASA's Observatorium's Gallery, a collection of the best in NASA photography and offering full-color, downloadable imagery, now has an array of new space photos online.

You'll like what you'll see in the Gallery. From pictures of the universe and all its unique inhabitants to exciting NASA photography of missions past and present, Gallery gives web surfers an up-close look at Earth and space science.

NASA's Observatorium homepage address at http://observe.ivv.nasa.gov. Contact:

Scott Gillespie
NASA's Observatorium
info@observe.ivv.nasa.gov

Frank Davis has LarKen blank boards for Disk Operating Systems. He also has blank LarKen RAM disk boards.

--==GATOR==--

John J. (Jay) Shepard III, wrote:

Denny, Do not count yourself out till we hear from Jack and what the magicians at Computer Classics did with his Larken. If they turned their magic, do not expect that I have it till I check what I rcvd, which I haven't. And if I have it, do not expect to rcv it just by asking for it, till those who started this have sorted out all that had been set in place, which included Frank Davis and Fwd Computing. When all that is resolved you will maintain your position behind or beside Jack ;

J. Shepard

Thanks for the email about the Larken. I still haven't gotten mine back from Computer Classics. I did receive an email from Dan a couple weeks ago stating he was just beginning to work on it. I will keep you posted.

Take care,

Jack Boatwright

Hello Abed,

Well, here it is, the first day of June and I am sending you a check in the amount of $49.71. I have shipped the last box to Iowa and Jack Boatwright will be picking up whatever is left.

I want to thank you, Bob, and the rest of T/SNUG members who donated the funds to get the inventory moved. I hope that a lot of it finds new homes with someone who can make good use of it.

If I can be of assistance to you or anyone else, do not hesitate to ask.

All of the masters for ant software that I had the rights to are now with jack Boatwright. I think that he will be willing to distribute whatever folks need or want if he has it. Thanks again, Sincerely,

Rod Gowen

Hi Abed & Bob,

I picked up what Rod thinks is the last of the items on Friday. Not much really, cables and RF boxes mostly. I did purchase another fellow's TS collection that day too. I now have quite a lot of TS items, but quite a bit went out after the list in ZQA! was published. I hope to have an updated list for you in a month or so.

Take care, Jack

Dear Al,

Here is a letter, I sent to Bob Swoger today, explaining where I'm at in this project and what kind of information is necessary to complete it.

I'm wondering, if you can't read it and maybe also have some helpful ideas :

Dear Bob :

I have tried several software patches here, trying to isolate the difficulty, MaxCom has SENDing files to modern computers.

According to the series of decisions, made by the CODE, the only thing preventing a normal exit is the receipt of a "6" in response to a "4" sent at the end of the last block.

So, would you review what to expect from current communications software, at the end of the last block SENT?

The APPLE people are dragging their feet on this information, so maybe the Timex people have the answer !

K E E P O N T I M E X ;n

So, Al, what do you think ?

David Lassov

Abed,

I really need help on this !

In Larry's doc on MaxCom, he gives all the control characters, SENT by MaxCom to the remote caller. The list is about 30 characters long, and he says MaxCom responds to ^Q, ^S, ^G, and ^\ .

Well, it looks, as though he sends out an EOT (^D) and waits for an incoming ACK (^F) . The 04 for the EOT signals, that no blocks of 128 bytes remain to be sent, and the 06 for the ACK is used throughout the CODE, to indicate a successful transmission, (then PRINT a '+' !) He also PRINTs a '-', upon receipt of a NAK, instead !

So, it looks, as though MaxCom responds to 06, or ^f, or ACK, and 21, or ^u, or NAK.

Now, the APPLE uses Proterm communication software, and it returns an error after the last block is received. It then hangs, and so does the Timex.

And, **the error** is "got noise after EOT (1/2688)"

That 1/2688 is interpreted, as follows: "1" stands for "one" error, which was detected at byte #2688, which is always "a 128-byte boundary."

So, everything seems to be working, OK. Just why doesn't Proterm send an ACK, in response to the final EOT ?

David E. Lassov: Sysop, SOL BBS @ 520-882-0388 (data)
520-882-3972 (voice) emanon@azstarnet.com (email)
2590 N. Jordan DR
Tucson AZ 85745-1132

Hello Abed,

Great News!!! I can now READ my own mail and other printed material!

It has been a long time coming, but I finally had to mortgage my soul and pay the price to be able to read my own material.

I am enclosing a photocopy of the spec sheet for the machine that I now have working for me. At a price of $3700.00 it was not cheap, but it will be all mine as soon as I pay the bank off.

I would like to be put back on the mailing list for ZQA! if at all possible. I can now scan it in and hear it in a matter of minutes. If I need to send you a subscription fee please let me know.

I hope that we are all straight on the inventory that I shipped. I know that Jack Boatwright has been shipping a lot of items out to folks who saw the info in ZQA! as well as on the net. If we are not straight on it, please let me know

I will reiterate that I am still available if someone needs help in any way that I am capable of giving it.

Thanks again for all of the help and your support in the past.

Thanks also to all those who gave of their time and money to get the TS inventory moved and distributed. Pass them along. Sincerely,

Rod Gowen

I'm looking for a working one, pref. with printer + modem, but I don't think I can be very choosy at this point. ;) If you or any of your member's group (TSNUG) have any they're willing to part with, please let me know, or let them know I'm looking to buy. Thanks!

Louis Florit expressweb.com<

florit@grep.unixville.com

I will put an ad in the next newsletter. If you like, send me your snail mail address for those who don't use email.

ZX-81

Thanks Abed! My snail mail is

Louis Florit

5445 SW 150 Place

Miami, FL 33185

My pager number is (305)478-3278

From: Alvin Albrecht <aralbrec@concentric.net>

Newsgroups: comp.sys.sinclair Date: Wednesday, July 01, 1998 Subject: Re: Timex /Sinclair Users groups and suppliers

Michael Owazany wrote: > Anyone know of any Timex Users groups in the states these days. Or any one who still sells the machines and software

Yes. Groups: TSNUG: (Timex Sinclair North America User Group) Email: AKahale /at/ compuserve <dot> com. LIST (Long Island Sinclar Timex Group) Email bmalloy <at> idt (dot) net. A few others as well. You can find the addresses in the TSNUG newsletter.

Vendors: FWD Computing Email: fdavis !at! iquest ?dot? net. Also ask at TSNUG as the inventory of another vendor, RMG, has been transferred to them. Alvin

Josh Payne

Joshpayne@bigfoot.com

Hello Abed,

I still have a lot of stuff (I bought another collection a couple weeks ago from someone Rod knew).

I still haven't gotten my Larken back from Dan Elliott. Sheesh! Oh well, in the latest bunch of stuff I got an Oliger with LKDOS, 2 more 2068s, a QL, a 1000 that is really loaded and a whole bunch of odds and ends.

I've been selling the extra stuff from my personal collection as well as sending the RMG items for shipping costs. Not getting rich, but having a lot of fun and meeting new people.

I hope to have you a revised list of the RMG items before the next issue of ZQA!

Jack Boatwright

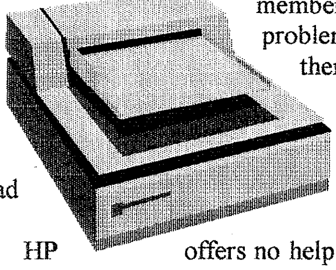
Fire in the Hole

HP 500/C printer *by Robert Swoger*

Recently my wife's HP 500C printer started to smell bad and as it was in a low light area of the room, a small flame was visible in the right hand side of the paper tray. Shortly after that moment, the printer went 'lights out'.

I opened the printer to find that a small drop of ink had fallen onto the printer logic board and caused a small fire just under the print cartridge parking area. It burned up two chip capacitors as well as the PC board under them.

I mentioned the problem at the Glenside Color Computer Club meeting and was surprised that a few of the members present knew of the problem first hand as two of them had experienced the same problem and had to buy new printers (not HPs). I wish they had told me sooner!



I found that HP offers no help in providing schematics or technical support for their products. I shall probably stay away from them next time as they told me to forget about trying to fix it and buy a new printer.

To save the HP printer that you might have, open your printer and place a plastic or cardboard ink shield over the PC board just under the area where the print cartridge travels, especially under the print cartridge parking area. This area is readily seen by looking for the two banks of four diodes in the right top corner of the PC board.

Fix it BEFORE it happens to you and - GOOD LUCK! ---==GATOR==--

Sinclair E-Mail List

Albrecht, Alvin	aralbrec@concentric.net
Anderson, Paul	panderson2@home.com
Anson, Gerald	jerrya@aztec.asu.edu
Barker Robin	Robin@di-ren.demon.co.uk
Bennett, Dave	dbennett@epix.net
Boatwright, Jack	jboatno4@outlawnet.com
Boehm, Al	boehm@ziplink.net
Boehm, Bill	boehm@plh.af.mil
C. A. T. S.	mf0002@epfl2.epflbalto.org
Cable, Bill	bcable@triton.coat.com
Chambers, George	gfchamb@pathcom.com
Collins, Bill	bcollins@home.ifx.net
Cottrell, Les	jacottre@gte.net
Cruz-Figueroa, Jaime	cruzfiguer@aol.com
Dansby, Andrew	adansby@atlantic.net
Davis, Frank	fdavis@iquest.net
England, William	wengland@iname.com

Feng, Al	alfeng@juno.com
Florit, Louis	florit@grep.unixville.com
Fink, Mike	domino.cubes@excelsior.net
Franke, John	j.m.franke@larc.nasa.gov
Ganger, Gary	gangerg@dma.org
Gillespie, Doug	aa431@cleveland.freenet.edu
Harbit, Ken	krh03@csufresno.edu
Henderlight, Mike	mikehend@microsoft.com
Henn, Fred	oranur@juno.com
Hunkins, James	jdhunki@ibm.net
Impellizerri, John	jimpellizerri@compuserve.com
Jaap, Matthias	Matthias_Jaap@hhs.hh.schule.de
Jonas, Mike	mjonas@bbn.com
Jones, Terry	tjones@iname.com
Kaczor, Jon	75363.1127@compuserve.com
Kahale, Abed	akahale@compuserve.com
Kealy, Harriet Joan	hjkealy@ms1.hilconet.com
Kingsley, Ed	elk4@aol.com
König, Urs	urs.koenig@agrodata.ch
KurtK7	kurtk7@aol.com
Kwitkowski, Phillip	kwit47@aol.com
Lancaster, Garry	dharkhig@delphi.com
Lassov, David	emanon@azstarnet.com
Lebowitz, Dave	dkl@dpliv.com
Lessenberry, Gary	glessenb@usr.com
Liebert-Adelt, Peter	p.liebert@t-online.de
Malloy, Bob	74776.1161@compuserve.com
McKelvey, William	mckelveyw@delphi.com
Merz, Jochen	jochenmerz@j-m-s.com
Miller, Seymour	seymil@delphi.com
Muth, Bob	bobkeeper1@aol.com
Norton, Gary	gnorton@world.std.com
Parrish, Gil	107765.1161@compuserve.com
Pashtoon, Nazir	napware@juno.com
Pazmino, John	john.pazmino@moondog.com
Perry, Russ Jr	slapdash@enteract.com
Rigter, Wilf	rigter@cafe.net
Rish John	74601.1535@compuserve.com
Shepard, Jay	jshepard@netins.net
Simon, Thomas	73177.333@compuserve.com
Skapinski, Tom	tskapins@juno.com
Smith, Dennis	denny.smith@juno.com
Swenson, Tim	swensontc@geocities.com
Swenson, Tim	swensont@sirclive.csd.sgi.com
Swentko, Wally	wswentko@maroon.tc.umn.edu
Swoger, Robert	ceng108@email.mot.com
Taylor, Jeff	jetaylor@spar.ca
Thoresen, Jeff	74200.257@compuserve.com
Walterman, Don	walterm@ix.netcom.com
Washington, Barry	mf0002@epfl2.epflbalto.org

Puzzle

Gil Parrish

Perhaps someone can help me with identification of a TV/monitor interface that appears to be ZX/TS related. The story goes that I was in a thrift store and found a PC8300. If you don't know, a PC8300 is a Hong Kong "clone" of a ZX-81 or TS1000, with many similarities (like an identical expansion edge card), but also with some different and additional features. (I did a piece on writing music for one in the Summer '95 issue of ZXir QLive Alive!). Anyway, sometime later I was in a different part of the same store and ran across the video interface in question, so that interface may or may not have anything to do with the PC8300.

It has a non-shiny creme plastic case (the PC8300 has a shiny creme case) with four diagonal stripes on the front, being (top to bottom) red, blue, yellow and green. (The PC8300 is marked with green of about the same shade, but none of the other colors.) It has a ZX/TS-style expansion connector with passthrough edge card, and a short (maybe 6 to 7 inch) hard-wired cable on the right side (as you would view it from the computer keyboard) with an RCA plug at the end. On the left side, it has two RCA jacks, labeled "TV" and "Monitor", and (if you open the case) the TV one is connected to an RF modulator box not unlike what you find on a ZX/TS motherboard. There is no marking on the outside of the case, and about the only marking on the board inside is "LAMBDA".

Now, my first guess was that this added a monitor port to the PC8300 by attaching to the TV output. But then I remembered, the PC8300 already HAS a monitor port, as well as a TV port (Channel 3). I then thought maybe it was intended to add a monitor port to the ZX/TS. But the expansion edge card on a ZX/TS is (as you face the keyboard) to the far right, so that right-hand cable on the interface couldn't attach to anything. (The TS2068, by the way, also has its expansion edge card at the far right, and also already has both a TV and a monitor port. While the TS2068 does have three short stripes on its primarily-silver case in the colors of red, green and blue, it has nothing in creme or yellow.) The PC8300 has its expansion edge card in the middle, with the monitor port to the right and the TV port to the left, so the interface could fit that edge card and have its RCA plug in the monitor port. But again, why? The PC8300 already has a TV and monitor port.

Despite the lack of sense, I tried attaching the interface to the expansion and monitor ports of the PC8300. My monitor (hooked to the monitor output jack of the interface with an RCA-to-RCA cable) showed something that might have been a normal picture, only with vertical and/or horizontal hold totally misadjusted. No amount of adjustment brought the picture in. On the TV, I got nothing until I hit UHF channel 35, and then I got the same kind of twisted, unadjustable picture I got on the monitor.

I can speculate a lot here. The device could be some kind of faux-color adapter (hence the stripes), but this particular interface is defective. Or maybe it was intended for the European Spectrum or other related model (how would I know, when I've never even seen a Spectrum in person?) and won't work on North American systems. Or possibly it was intended to modify the PC8300 to European (PAL or SECAM) use. Or perhaps I'm simply missing the point as to its purpose or usage on the PC8300. Anyway, if someone knows, drop me an e-mail at

107765.1161@compuserve.com

and take me out of my puzzlement.

My wild guess is that it was made for viewing the Spectrum on US video monitors. Editor

FROM THE CHAIRMAN'S DISK

Donald S. Lambert

What a summer! Hot or hotter than usual and then a dry spell for a while and then too much rain for some areas. Living in Auburn, IN gives us some protection from the storms since we are the right distance and direction from Lake Michigan to be in the area too far from the storms that the lake generate and too near from the lake for the other storms to really affect us with the extreme rain (or snow) fall that others get. And since all our power comes in from other areas we do get an occasional power losses mostly caused by excessive loads during the really hot weather. That really affects those that have a sump pump. We don't have a basement so no worry about that.

Masakos my wife was watching TV one day and she tried to change from her soap to QVC (a shopping station on cable) by way of the FLASH button and it didn't change stations. She tried the up and Down button and while the screen flashed as it did for the FLASH button it didn't change stations. So of course I was called to the rescue and I checked the other TV and it worked properly so I made an immediate conclusion that the AAA batteries in the remote were weak. However ! with the original batteries it would get the proper station if you punched in the station number. I tried another set of batteries and still the same and I tested the batteries and still good so now what? Either the TV or the remote was bad. I took the remote and went to the local store where we bought the TV and the remote tested good (it put out an infrared beam) and a set of the same make (Zenith) was on in the display so I found that remote was working for sure.

The service man suggested that maybe the power outage had done it. Why wouldn't it show up right after the outage? So back home I read the manual and did a from scratch start up. Which was to have the TV find out which channels had a program. While I was at it I reset the clock that showed on the screen when you changed stations. After that the remote operated correctly. Nowhere in the manual does it mention what to do in case of a power down situation. In my eyes that TV had a case of loss of short time memory or a case of Alzheimer's disease. What connection to the T/S computers; Just pointing out there is so much that isn't manuals covered in the manuals.

The May issue of NUTS and VOLTS magazine in Tech Forumn department of the magazine was the request for informatiom. I have an old Z88 made in Scotland that works. It has new AA batteries. "How do I connect it to my IBM 486 or my PowerMac 7200?, and with what cables in order to download notes of ASCII type. Do I need software and where can I get it? David McGuire." I sent in information about the cables and software that Frank Davis had for sale and NUTS and VOLTS magazine printed my reply in the June issue. They stated that the first reply printed in the answer column of that department would get a check of $25. My wife laughed "oh yeah!" The check arrived on August 15th. Now what will I buy with it?

When I checked the price of dishwashers at Sears I drifted over to the computer section TV/Internet deal. A poor man's way of going on line. This one had a printer port and a hard drive but no floppy drive. Complete with printers keyboard and interface it ran about $400 new and only required to pay the $20 monthly fee to go on line. I was thinking of hooking one up to my AMDEK COLOR-I monitor of which I have two. But then I remembered that I could probably buy an IBM compatible used computer for less than that and have the ability to do so much more. But then I would have to relearn how to use a computer. And I still have the Z88 to Conquer. I have lived this long without being on line so I can live some more.

I do have a Z88 and a modem for the Z88 so I could go that route to go on line. BUT! just how does one do it?

Can someone give me a step by step guide as to how to do it?

External battery case for the Z88 — I wanted to have a way to keep a Z88 alive for extended periods of time and the internal AA batteries don't last long. By that I mean if the Z88 is in a hanging file folder for half a year at a time. I could use alkaline batteries but then they do expire given enough time. I thought of rechargeable batteries the first choice is NICAD BUT! the NICAD have a tendency to discharge while sitting on the shelf. The literature states that a NICAD will self discharge in about a month. So they are out. But I have had good results with RAYOVAC RENEWALS in the three quartz clocks we have. They each use one AA battery and I swap them out and recharge them periodically and have for several years.

So I collected the necessary parts and the 'D' sized batteries. I managed to find the batteries on sale so that helped so were the coupons that I redeemed to even get more discount. I have 6 batteries in series to power the Z88. O.K. I know that the total theoretical voltage of 6 batteries is 9.9 volts. So I put seven lN4007 diodes (why lN4007 diodes that was what was in the diode bulk package at Radio Shack) in series with the power and have a DIP switch across each so that I can short out the diode if want a higher voltage. I also found out that with no load on the circuit that I get a higher voltage than what I expected. I did not charge the batteries before I tested the battery box that built. The batteries in series give 9.42 volts. I tested the computer in the sleep mode (turned off) and with it powered up. I used a digital multimeter. The following charts gives the values that I got and the action of the Z88.

No Load 6.94	**No Load 7.28**	**No Load 7.29**
Sleep 3.54	Sleep 5.82	Sleep 5.95
Z88 ON	Z88 ON 3.98	Z88 ON 3.95
Doesn't Power Up	Gives BAT low	Gives BAT low
No Load 7.62	**No Load 7.97**	**No Load 8.31**
Sleep 6.34	Sleep 7.08	Sleep 7.56
Z88 ON 4.87	Z88 ON 5.63	Z88 ON 6.30
Gives BAT low	Gives BAT low	Full operation

According to some literature that I received from RAYOVAC the RENEWAL batteries give more recharge cycles if they aren't discharged too deeply. One place mentions that they shouldn't be discharged below 1.3 volts or a total of (6 X 1.3) 7.8 volts. I will try to aim for a total of 8.4 or 1.4 per battery to be sure. Even so they should give a long life compared to AA batteries. 0/0

QL Hacker's Journal

Tim Swenson

#28 **July 1998**

The QL Hacker's Journal (QHJ) is published by Tim Swenson as a service to the QL Community. The QHJ is freely distributable. Past issues are available on disk, via e-mail, or via the Anon-FTP server, garbo.uwasa.fi. The QHJ is always on the look out for article submissions.

QL Hacker's Journal
c/o Tim Swenson
2455 Medallion Dr.
Union City, CA 94587
swensontc@geocities.com
http://www.geocities.com/SilconValley/Pines/5865/
Note New Mailing Address

Editors' Forum

I have not received much feedback on the QLiberator Source Book idea I mentioned in the last issue, and I have received basically no "helpful hints". So, my plan is to write what I can, put it out on the Net, and see what feedback comes from it. If I make any mistakes, I'm sure there are many willing to point out my failings :-). Sometimes the best way to get an answer to a question is to give a wrong answer, then many will pop up to give the real answer. I'll probably put in questions that I have in the hope that they too will get answered.

A part of the material going into the Source Book is coming from stuff I'm writing for this newsletter. I have taken the article on adding Config Blocks to QLib program and expanded it quite a bit. A couple of articles from this issue will also go in. I also plan to have sections that touch on the various SuperBasic extensions and programming aids available to the SuperBasic programmers.

In other news, besides having past issues of the QHJ available on my web page, I've added a number of articles that I've written for other newsletters. Check the link from the main page.

Structured SuperBasic 2.6

Structured SuperBasic is a utility that has been printed a couple of times in this newsletter. I have recently dusted the program off and completed a new version. SSB 2.6 has all of the functionality of SSB 2.5, but I have added a lot of error checking, fixed a few bugs, added Config Block support, Command Line support, Environment Variable support, and compiled it with QLiberator. The manual has been expanded from 5 pages to 16 pages. The whole package has been zipped and should be available your local QL BBS and the Internet. It can be downloaded off my web page at http://www.geocities.com/Silicon Valley/Pines/5865/.

For those that don't remember what SSB is, it is a filter program that takes SuperBasic code, written in the form for SSB and converts it to full SuperBasic code. SSB allows for no line numbers, blank lines between code segments, conditional compilation, Include files, and other features. SSB allows for more readable and maintainable code. Download the package, unzip it, and give it a try.

Revision Control System (RCS)

Whether it's source code or system configuration files, it's nice to be able to keep track of changes made to files over time. The Revision Control System is a collection of programs that keep track of different versions (revisions) of text files. A special file is created by RCS that contains information about the changes in a file and allow the user to get back to any revision of the file.

RCS was designed primarily for programmers to keep track of what changes were made to various source files. It was derived from an earlier Source Code Control System (SCCS) and was first developed for the UNIX operating system.

Since RCS came from a UNIX background, it understands the concepts of different users accessing the same files and allows "checking out" files and locking them from being changed by other users. RCS can be used by a number of programmers all working on the same code. It keeps track of who has what file "checked out" and who makes what changes.

Although most QL programmers program by themselves, RCS can help the QL programmer bring a form of discipline to their programming. This is especially useful for what can be called "full production" programs, commercial or Freeware. Given the long life of QL programs (I'm still using programs written more than 10 years ago), RCS can keep track of what changes were made from revision to revision and why. A programmer can keep make a rule that each RCS revision be a bug fix and the reasons for the fix can be logged as part of the revision history.

Enough blathering on, now to jump right into what RCS is and how it works.

Original File - This is the text or source code file that you want to keep track of. It can be any text file.

Revision File - This is a single file that contains the original text/source file and all of the revision history. It has the same name as the original file except that a ',v' is appended to the end. If the original file is 'test_c', then the revision file is called 'test_c,v'. A revision file keeps track of only one original test/source file. If you have 3 source code files that make up a single executable, then RCS will have three revision fills.

Check-In - All revisions to a file are put into the revision file by checking them in. The program 'ci' is used to do this. 'ci' is used to either create a new revision file or to check-in a new revision into an existing revision file. Once a file has been checked into the revision file, it is deleted.

Check-Out - When you want to edit a text/source file that has been checked in to the revision file, you use 'co' (check-out) to extract the text/source file from the revision file.

The whole process of using RCS is basically checking-in and checking-out the same file. RCS is not aware of time, so there is no need to check-in a text file when you have finished a programming session. You really only need to check-in a file when you have made all of the edits you need. If you are fixing a bug in a program, you would only check-in the file when you have fixed it. No need to keep track of working copies of the files.

To show you how RCS is used, I've written a short Structured SuperBasic program:

```
      OPEN #3,con_100x100a100x100
      CLS #3
      INPUT "Enter The Name of a File
:";file$
      OPEN_IN #4,file$
      REPeat loop
         INPUT #4,in$
         IF EOF(#4) THEN EXIT loop
         PRINT #3,in$
      END REPeat loop
      CLOSE #4
      CLOSE #3
```

I stored this program in the file 'readfile_ssb'. I then checked it in to RCS using the following command:

```
exec ci;"readfile_ssb"
```

'ci' then asks me for my initials. Since the QL does not have the concept of multiple users (and therefore user names), a persons initials are used to keep track of who has checked out a file and who has made what changes to that file. 'ci' then asks for a Description. The Description is where the "why" of the file change is kept track of. If you are using RCS to keep track of bug fixes for a program, the Description section would be used to mention the bug, what caused it, and how it was fixed. I do not believe there is a limit on how long the description can be. The Description is ended by a line with just a period on it. 'readfile_ssb' is now checked-in to the RCS revision file and is deleted.

Now to get the file back to edit it, I ran the following command:

```
exec co;"-l readfile_ssb"
```

I need to tell 'co' what file I want, but I also have to tell 'co' that I want to edit the file. If I ran 'co' without the "-l" option, 'readfile_ssb' would have been created, but it would not have been "checked-out" from the revision file, and when I went to check it in, RCS would not have allowed it. Under UNIX, 'co' would create a 'read-only' copy of the file. 'co' asked for my initials.

Since the same person who checks out a file is the only person the can check in a file, be sure to remember what initials you used.

'readfile_ssb' is now extracted from the RCS file and the revision file is changed to show that the file is checked out.

Once I am done editing the file, I can then check in the file using 'ci'. Again I am asked for my initials. The file is now checked back into the revision file and deleted. If you copied the file before checking it back in and you have edited this file, you can not incorporate those changes into the revision file since the file is 'officially' checked in and can not be edited.

RCS uses revision numbers to keep track of the different times a file is checked-in. The first revision is 1.1, then 1.2, then 1.3 and so on. Let's say that I have edited my file 4 times, I'm now at revision 1.5 and want to recall revision 1.3. I would run the following command:

```
exec co;"-r 1.3 readfile_ssb"
```

'co' would then dig though the revision history and generate version 1.3 of 'readfile_ssb'.

RCS comes with more than just the 'ci' and 'co' programs. 'rcsdiff' is used to compare a working "checked-out" file with the version still in the revision file. 'rlog' is used to view the revision file.
Below is the output of 'rlog' from my example session:

```
RCS file: readfile_ssb,v
Working file: readfile_ssb
head: 1.2
branch:
locks: strict
access list:
symbolic names:
comment leader: "# "
keyword substitution: kv
total revisions: 2;selected revisions: 2
description:
Creation of RCS Archive
----------------------------
revision 1.2 date: 1998/06/11 20:41:01; author: tcs; state:
Exp; lines: +2 -2
This is a second version of this program.
----------------------------
revision 1.1 date: 1998/06/11 20:19:34; author: tcs; state:
Exp; Initial revision
```

RCS is a lot more complicated that I've mentioned here. It, and SCCS, is fully documented in the book "Applying RCS and SCCS" by Bolinger & Bronson, published by O'Reilly & Associates. RCS for the QL is available from most Freeware sources. The QL RCS distribution does not come with the 'diff' program which it requires. 'diff' is available as part of the C68 distribution.

RCS can not be used with regular SuperBasic programming, as RCS would see any new line numbers as a change in the file. If you loaded a file into SuperBasic, did only a line renumber, saved it, and then check it back into RCS, RCS would any line with new line number as having changed. RCS works well with Structured SuperBasic as it has no line numbers. It also works well with C, Forth, Pascal, etc.

Environment Variables

The concept of Environment Variables comes from the Unix world. They are used slightly in MS-DOS, but not at all to the same extent as UNIX. For the QDOS world, the file ENV_BIN provides a number of extensions that allow the use of Environment Variables. Essentially an environment variable is a variable that can be "seen" by executable programs. In SuperBasic we can set up all kinds of variables, but if we execute a program from SuperBasic, these programs can not "see" these variables and get data from them.

The purpose of environment variables is to change the configuration of a program. They function like Config Blocks, but don't require running a program to make the change. Let's take a quick look at how we can change the behavior of programs. There are five different ways of doing this:

1) User Intervention. This is where the user uses a menu or answers queries from the program.

2) Config Blocks. This feature is pretty unique to

QDOS, but it allows the user to change default options without having to know how to edit a binary file.

3) Configuration File. This is a separate file that the program reads to determine how to set defaults and run.

4) Command line Arguments. Instead of the program querying the user for information, the user types it in when they execute the program.

5) Environment Variables. The user sets a variable that is then read by the program and changes its default settings.

Each of the options have their own place and their own benefits and faults. Some are more permanent, like Config Blocks and Config files, while some are very short lived, like User Intervention and Command line Arguments. Environment Variables are in between as they can be set in a BOOT program, but they can be changed by just typing in a new command.

The ENV_BIN file comes with 4 extensions. They are:

SETENV	- Defines an environment variable.
ENV_LIST	- Lists all defined environment variables.
ENV_DEL	- Deletes an environment variable.
GETENV$	- Gets the value of an environment variable.

The two key commands are SETENV and GETENV$. SETENV is used like this.

```
SETENV "VARIABLE=value"
```

SETENV takes a string argument of the type "XXXXX=YYYYY" where XXXXX and YYYYY are two strings separated by an equal sign. Any space before the equal sign is treated as a part of XXXXX and any space after the equal sign is treated as a part of YYYYY. The case of each string is important as "VARIABLE" is different from "variable". By convention, upper case is used for variables.

The SETENV is done either in a BOOT or setup program or by the user. The command for an executable to get the contents of an environment variable is GETENV$. The command is used like this:

```
a$ = GETENV$("VARIABLE")
```

In this case a$ will be assigned the value of "value". This comes from our previous SETENV command above. If the Environment Variable "VARIABLE" is not set (does not exist) then the GETENV$ would return a Null string ("").

Now I did say that executables use GETENV$ and not SuperBasic programs. Since variables are already used in SuperBasic, we would not gain much in using environment variables. There the commands are use is in compiled SuperBasic programs, which are executables.

I see the purpose of using Environment Variables as adding to the flexibility of programs that use Config Blocks. Both Config Blocks and Environment Variables are really designed to change default program settings. User Intervention and Command Line Arguments are designed to tell the program what the data is. Using environment variables allows the user the ability of making a temporary change to the default options of a program, without having to go through the trouble of using "config". Environment variables would be used to change a setting for a single session and that's it.

Getting Config Blocks and Environment Variables to work together is not difficult. The program would first get it's default settings from the Config Block. It would then check to see if there are any Environment Variables set. If there are, then the Environment Variable settings would override the Config Block settings.

Working Directory

As we all know, QDOS did not come with the concept of directories and subdirectories. A number of extensions to QDOS are available to help create the working concepts of directories. When using ED drives, I found the Path (PTH) extensions useful because they would search a list of subdirectories looking for a particular executable.

Now, I've never used a QL with a hard disk, so I'm not too experienced with using the tools for handling directories, plus my copies of some of the good articles dealing with this subject are not available to me. So, I've been doing some thinking on the matter, especially since I have been writing a Freeware program and I want to make it as versatile as possible.

The one issue that I've been pondering over is telling a program where there data file are at. This is called a Working Directory. It is a directory where the data files are at and any newly created files will also go.

I know that DATA_USE is designed for this purpose, but I do not want to type in a new DATA_USE before each program I execute. I could set up a short little SuperBasic program to set up everything, but I want to EXEC a program not LRUN it. Of course, using a front end like QASCADE, this point is mute, since the user is removed from the details of executing the program. With PTH_ setup, I don't need to set PROG_USE, because PTH_ finds the executable. I was looking a way of having the executable know where the Working Directory is.

What I came up with is this: Have an item in the Config Block for Working Directory. The contents would be something like this, "WIN1_PROG_DATA_". This string would be appended to the front of each file name used in the program. This does mean that the user would not be able to type in a new device name, such as "FLP1_FILE_EXT". Initially, the Config Block entry for Working Directory would be the null string (empty), so that a user would enter "FLP1_FILE_EXT". Once the user has created a working environment, the Config Block can be changed to what the user wants.

If this feature is linked to Environment Variables, it would become more powerful. By setting an Environment Variable for the program, the user could temporarily override the Config Block and change the Working Directory. If the Config Block was changed to a Working Directory and the user needed to use the floppy for a few times, the user would set the Environment Variable to "flp1_" or to the null string ("") and then set DATA_USE to flp1_. If they did not want to change the default Config Block, a BOOT program could set the Environment Variable and it would be active during the whole computing session. If different programs used different Environment Variable names for their Working Directory, then all Working Directories could be set up in the BOOT program. Something that could not be accomplished with just DATA_USE.

Since my experience in this area is kind of light, the more experienced QLers may want to take this all with a grain of salt. I prefer to think of it as a way that I would prefer to organize things and others may have a different way.

The reason I am bringing up this idea is for programmers to consider the concept and add it to their programs. The option is not very difficult to implement, but it would a nice feature to the program. The whole feature would take up no more than 10-20 extra lines of code to a program. By setting the default value of Working Directory to nothing, the feature is essentially turned off and it does not impede the user that does not want use a Working Directory.

Case Statement Implementation

When I recently was working on SSB 2.6, I was using a SuperBasic implementation of a CASE structure for the core of the program. The deeper the structure got, the harder it was to read and understand. I starting thinking of using another way of implementing a CASE structure.

For those that don't know what a CASE structure or statement is, it is essentially the same structure as a SuperBasic SELECT ON statement, but it is not limited to numbers. Plus the general idea of a CASE structure is to have a number of possible logic statements with an action for all cases that do not fit one of the logic statements. In a generalized CASE structure not all of the comparisons must be based on the same data (string, number, etc), but can vary.

Traditionally, a way of creating the CASE structure is to have a number of nested IF..THEN..ELSE statements, with the final ELSE handling the default value. An example would be something like this: you are reading in text from a file. You want to handle the following cases: a line starts with a ##, or a line starts with a **, a line starts with a period (.). If none of these cases, then the line is passed to the output file.

Using nested IF..THEN..ELSE statements the pseudo code would look something like this:

```
IF line starts with ## THEN
  ELSE
   IF line starts with ** THEN
    ELSE
      IF line starts with a period THEN
        ELSE
           pass to output file
        END IF
     END IF
  END IF
```

If you start having more than just a hand full of possible cases, the structure can get long and difficult to read.

SuperBasic allows the use of NEXT in conjunction with a REPEAT statement. This means that when the NEXT is reached, the rest of the REPEAT loop is skipped and the processing goes onto the next interaction of the REPEAT loop. Using NEXT in this manner allows for the creation of a different implementation of a CASE structure. Below is an example of the two implementations listed side-by-side.

```
REPeat loop            REPeat loop
 IF .... THEN           IF .... THEN
  ....                 ....
 ELSE                   NEXT loop
  IF .... THEN          END IF
   ....              IF .... THEN
    ELSE                  ....
     IF .... THEN          NEXT loop
      ....             END IF
     ELSE                IF .... THEN
      default              ....
     END IF               NEXT loop
    END IF               END IF
   END IF                default
  END REPeat loop          END REPeat loop
```

Using the REPEAT..NEXT..END version may not seem as clean as the "classical" IF..THEN..ELSE structure, but I think it is cleaner when it come to reading and debugging. What I am looking for from readers is to ponder over any downsides from using the REPEAT..NEXT structure. I can't think of any logical problems with using this structure over the classical one. If you know of any, please send me a note. I'll add the productive comments in the next issue.

Creating Loadable Extensions Using QLIB

One of the things that has always amazed me about the QL was the ability to load a binary file and have a bunch of new keyboards available in SuperBasic. In most computers that had Basic built in, the language was static and had no way to extend itself. Other languages (like C, Fortran, or Pascal) used libraries of functions and procedures to extend the capability of the library.

The first major LOADABLE extenuation to the QL was ToolKit. From then on the term toolkit has been used in reference to LOADABLE extensions. Popular toolkits are ToolKitII (TKII), DIY ToolKit, and DJToolKit.

I knew that the first of these toolkits were written in Assembly, but I did not know that they could be created by QLiberator. It seems that QDOS executables and extensions are real close in format and when compiled right, they can be interchangeable. This means that an executable can also be loaded as an extension. To figure out how to create a toolkit, I grabbed a simple function, QLiberator, and gave it a try. The function is included below:

```
10 REMark $$external
100 DEFine FuNction upper$(up$)
110  LOCal x, temp
120    FOR x = 1 TO LEN(up$)
130     temp = CODE(up$(x))
140     IF temp > 96 AND temp < 123 THEN
                up$(x)=CHR$(temp-32)
150    NEXT x
160    RETurn up$
170 END DEFine
```

The function takes any string and converts it to all upper case letters. The $$external is a compiler directive to QLiberator that tells it that the next function or procedure needs to be available outside of the executable. For each procedure or function that you want to turn into an extension, you would have to put the $$external directive in front of it. If I was to put an additional line in the program,

```
  180 PRINT upper$("This is a test")
```

then when I executed the program, line 180 would be executed. If I LRESPRed the program, the keyword upper$ would be made available.

When compiling the program it is a good idea to turn WINDS off, since the extension will have no channels open. Otherwise 3 channels will be opened for it, wasting them. To lower the size of the binary file, turning off NAMES and LINES might be a good idea. Note that the case of the function or procedure will be maintained in the extension. In my example, the name that will show up when entering EXTRAS is "upper$" (all lower case). If I defined the function a UPPER$, then "UPPER$" would show up in the EXTRAS command. By convention, extensions should be done in all upper case.

If you will be running the extension on a system that already has the QLib runtimes loaded, then compile the program without runtimes. If you don't know if the QLib runtimes will be available, compile it with the runtimes included. It is a good idea to compile both ways and let the user decide which one they need. The example program when compiled without the QLib runtimes was 594 bytes. With the QLib runtimes it was 11,146 bytes. The runtimes take up a fair bit of space.

If you load an extension that does not have the QLib runtimes on a system where the QLib runtimes are not loaded, you will not get an error message when you LRESPR the extension. When you call the extension is when the error will occur. The exact error message is:

Error "Runtimes Missing !"

Once you have compiled an extension, all that is needed is to LREPSR it and test it out. Remember that you can't LRESPR while any jobs, other than Job 0 (SuperBasic), is running.

ZipaDee_Bas 1.105

by Al Feng

Several years ago, Anthony Trice was kind enough to send me a copy of the ZIP utility which was originally ported to QDOS by Erik Slagter & Kai Uwe Rommel. Unfortunately, need is a strong motivator, and it wasn't until this Summer that I finally sat down to figure out how to implement the utility in its QDOS incarnation. ZuipaDee (1.105) is the long over due front end for the ZIP utility that is meant to complement the earlier DooDah_bas front end for UNZIP.

While more than one compression algorithm exists, the use of the earlier 'LZ77' algorithm allows a QDOS ZIPped file to be universally expanded by almost all versions of UNZIP. The most recent port of ZIP that I have (v3.0) was executed by Marco Holmer.

Using Zip

There are many reasons for ZIPping a file -- the two reasons that come to mind are for archiving (storage) and transmission (distribution). Of course, the file or program cannot be used until it has been UNZIPped back to its natural form.

ZIPping a file results in a new, smaller file which is suffixed with '_zip'. For example, the TurboQuill version of Quill (60614 bytes) would become Quill_zip (39833 bytes). The command line syntax for using the QDOS version of ZIP is:

```
EXEC_W win1_ZIP;"zipname win1_fileone"
```

where 'win1_' is where the ZIP program is located (the location is optional), 'zipname' is the name that you want for the ZIPped file, and 'win1_fileone' is the location and actual name of the file you want to ZIP.

You should note that, as with UNZIP, the command and argument(s) are separated by a semi-colon; and, the arguments are enclosed within quote marks.

It has been said that sometimes a little knowledge is a dangerous thing; and, this was the case with my attempts to implement ZIP and write the ZipaDee_bas program.

Being familiar with the DOS versions of the ZIP program I observed that the QDOS version lacked the '-a' switch for amending existing ZIP files.

This seemed like a great omission that would require including all the filenames to be incorporated into a ZIP file at the time of the creation of the ZIP file. The command line syntax would be:

```
EXEC_W win1_ZIP;"zipname win1_fileone flp2_filetwo"
```

where 'flp2_filetwo' might be the device and location of the second file, and so on.

Subsequent to writing the current version of the program I belatedly ascertained that the '-a' switch has been renamed '-u' (i.e., update); and, the command line syntax is:

```
EXEC_W win1_ZIP;"-u zipname win1_fileone"
```

My ignorance prevented the program from resulting in a more cumbersome implementation for the user; so, all's well that ends well.

Select Device

There are several device options. Pressing 'W' will select 'win1_'; pressing 'X' will select 'win2_'; pressing 'F' will select 'flp1'; pressing 'G' will select 'flp2_'; and, so on. Use the 'other' option if you want to key in a device name (e.g., 'rom1_').

If you are looking at the menu of filenames and want to return to the SELECT_DEVICE option, press '0' (zero); however, you should note that all pre-selected filenames will be lost.

Change Device

To expedite multi-source ZIPping, you can change to different devices by using a "CONTROL SHIFT single_letter" key combination. The following are supported (you can modify the LISTing if you choose).

```
CTRL SHIFT F == flp1_
CTRL SHIFT M == mdv1_
CTRL SHIFT N == ndk1_
CTRL SHIFT O == rom1_
CTRL SHIFT R == ram1_
CTRL SHIFT W == win1_
```

To move from flp1_ to flp2_ (for example), simply use the right_arrow key.

CONTROL LEFT key to move to the device designated near the right_arrow near the bottom of the screen if the device type differs from the currently accessed device.

Press '0' (zero) to access the SELECT_DEVICE menu if you want to start over.

Zipping A File

Presuming that the program has been properly typed in, if you want to ZIP a program, you simply have to press the key which corresponds to the alphanumeric character in the brackets '{ }' to the left of the filename. The file will be ZIPped and stored on your default DATA_USE device (usually FLP2_).

If you have more than 1 Meg of memory then you can employ "DATA_USE RAM2_"; but, don't forget to move the(se) ZIPped file(s) to a disk before quitting!

ABOUT THE LLISTing

ZipaDee is a modification of the DooDah_bas program LLISTing, and some meaningless statements may still be included.

You should note the REMarked statements (Line 140 & 150). You can UN-REMark these if you have more than 1 Meg of memory.

If you do not have a hard drive, then substitute "win1_" with "flp1_" or any other appropriate device in Line 150 & Line 160 and (especially) Line 1010.

There are some changes to the general program, but if you already have a functional copy of DooDah_bas, then you can just change the "Main" DEFine(d) PROCedure to the following (your line numbers may differ):

```
    8nn LET Zipfile$=Z$(k+1)
    8nn filename$=t$&W$&"_"&Z$(k+1)
    8nn DoWhat$=Zipfile$&" "&filename$
    8nn CLS#2
    8nn IF k<c THEN EXEC_W
win1_zip;DoWhat$
```

As you can see, this will only provide limited functionality by allowing only one file to be ZIPped at a time.

If you want the capability to ZIP more than one file, then you will probably want to input the LISTing below (also based originally on the DooDah_bas program, so you can save some keyboard entry).

Originally, a more complex structure for multiple ZIPs was being written, but in a moment of inspiration, I realized that it might be possible to simply extend the DoWhat$ string; and, that is the method used.

The files to be included can be verified prior to ZIPping by pressing the '/' key. The files you have selected will appear in the upper part of the screen.

You will be asked for a ZIP file name after you have selected all the files you want to ZIP.

The following listing provides an active menu of 57 filenames (three columns x 19 files). If the files which you want to ZIP are not visible in the menu or contained within a sub-directory, then you will want to transfer the files to another source medium (e.g., ram1_, if you have "extra" memory); or, modify the QLUTter_bas program to contain the MANE & MAIN procedures instead of one of QLUTter's current utility PROCedures.

Happy Trails,
And Computing, To You ...

```
40 REMark
*************************************
50 REMark *    ZIP-A-DEE 1.105b      *
60 REMark *       by Al Feng         *
70 REMark *     A ZIP FRONT END      *
80 REMark DATA_USE RAM1_: ********  SAVE
as ZipaDee_bas  *******
90 REMark COPY win1_zip TO ram8_zip
100 F$="_FLIST_imp": t$="win": S$="win": W$="1":
Do$=" "
110 pn=0: POKE 163890,0: MODE 0
120 WINDOW#2,512,256,0,0: PAPER#2,7:
BORDER#2,1,7: INK#2,0:CLS#2:
WINDOW#0,413,10,50,241: PAPER#0,0:
INK#0,7: WINDOW462,250,25,3:
OPEN#3,scr_458x200a27x48: u$=" ZipaDee
1.105 ": AT#2,21,4:
130 PRINT#2,u$;" by Al Feng "\TO 4;" @
1998  PLATYPUS Software ": PAUSE 30
140 PAPER 7: BORDER 1,7: CLS: CL: lne:
AT 0,7: PAPER 2: INK 7: PRINT"
SELECT_DEVICE ": cj
150 DEFine PROCedure CLSc: BLOCK
458,225,0,10,0: END DEFine
160 DEFine PROCedure CLSd:
BLOCK#2,330,10,83,240,0: END DEFine
170 DEFine PROCedure CLSe: BLOCK
458,190,0,45,pn: END DEFine
180 DEFine PROCedure CLSf: BLOCK
458,30,0,10,0: END DEFine
190 DEFine PROCedure Press: CLS#0:
INK#2,5: AT#2,24,30: PRINT#2,"Press
Any_Key to Continue":END DEFine
200 DEFine PROCedure sx: DIM Z$(60,32):
DELETE t$&W$&F$
210 OPEN_NEW#6,t$&W$&F$
220 DIR#6,t$&W$&"_": CLOSE#6
230 OPEN_IN#7,t$&W$&F$: FOR c=0 TO 60
240 IF EOF(#7) THEN EXIT c
250 INPUT#7,Z$(c): END FOR c: CLOSE#7:
c=c-1: END DEFine
260 DEFine PROCedure sw: IF W$<=8 THEN
g$=W$-1: IF W$>=1 THEN h$=g$+2
270 IF g$=0 AND t$="flp" THEN LET
t1$="ram": ELSE t1$="flp": END IF :
g1$=1
280 IF g$>0 THEN t1$=t$: g1$=W$-1: END
IF
290 END DEFine
300 DEFine PROCedure Rx: CLSf: INK#3,7:
PAPER#3,0: AT 2,3: PRINT
t$;W$;"_";: INK 7: PRINT Z$(0):Rx1:END
DEFine
310 DEFine PROCedure Rx1:FOR a=0 TO 18
320 IF a<1 THEN a=a+a
330 FOR n=1+a+a: AT#3,a,0:
PRINT#3,"{";CHR$(n+a+48);"} ";Z$(n+a+1)
340 FOR n=2+a+a: AT#3,a,26:
PRINT#3,"{";CHR$(n+a+48);"} ";Z$(n+a+1)
350 FOR n=3+a+a: AT#3,a,52:
PRINT#3,"{";CHR$(n+a+48);"} ";Z$(n+a+1)
360 NEXT a: END FOR a: rx2: END DEFine
370 DEFine PROCedure rx2
380 STRIP#2,0: INK#2,7
390 BLOCK 2,13,18,235,2: BLOCK
2,12,436,236,2: BLOCK 458,1,0,235,2
400 BLOCK 18,12,0,236,0
410 BLOCK 20,12,438,236,0
420 BLOCK 416,12,20,236,0
430 AT#2,24,5: INK#2,7:
PRINT#2,CHR$(188);" ": AT#2,24,78:
```

```
PRINT#2,CHR$(189);" "
440 AT#2,24,8: INK#2,5:
PRINT#2,t1$&g1$;"_"TO 71;t$&h$;"_"
450 END DEFine
460 DEFine PROCedure Uu: sx: pn=0: CLSe:
Uv: END DEFine
470 DEFine PROCedure Uv: sw: Rx: k3: pk:
END DEFine
480 DEFine PROCedure Uw: wx: k3: pk: END
DEFine
490 DEFine PROCedure pk: k=k-48
500 IF k=-21 THEN CLS#2: nd
510 IF k<-1 AND k<>-14 THEN Uw
520 IF k=-14 THEN Uw
530 IF k=68 THEN Uv
540 IF k=144 THEN IF W$>1 THEN W$=W$-1:
Uu: ELSE : iX: Uw
550 IF k=118 THEN t$="flp": a$=1: sx:
MAIN
560 IF k=125 THEN t$="mdv": a$=1: sx:
MAIN
570 IF k=126 THEN t$="ndk": a$=1: sx:
MAIN
580 IF k=127 THEN t$="rom": a$=1: sx:
MAIN
590 IF k=130 THEN t$="ram": a$=1: sx:
MAIN
600 IF k=135 THEN t$="win": a$=1: sx:
MAIN
610 IF k=146 AND t$="ram" THEN t$="flp":
LET a$=1: Uu
620 IF k=146 AND t$="win" THEN t$="flp":
LET a$=1: Uu
630 IF k=146 AND t$="flp" THEN t$="ram":
LET a$=1: Uu
640 IF k=152 THEN IF W$<2 THEN W$=W$+1:
Uu: ELSE : iX: Uw
650 IF k=156 THEN W$=W$+1: Uu
660 IF k=79 THEN CLS#2: lne: tre=3: WCh
670 IF k=0 THEN GO TO 140
680 IF k>=58 THEN Uw
690 END DEFine
700 DEFine PROCedure wx: STRIP#2,2:
INK#2,7: AT#2,24,29:     PRINT#2,"
ERROR" TO 49;"ERROR ": AT#2,24,36:
STRIP#2,0:PRINT#2," WRONG KEY";:
INK#2,5: PRINT#2,"!": STRIP#2,7:INK#2,0:
B5: CLSd: B5: END DEFine
710 DEFine PROCedure iX: AT#2,24,35:
PRINT#2,"invalid drive": B5:PAUSE 10:
END DEFine
720 DEFine PROCedure k3: REPeat key
730 k=CODE(INKEY$): IF k>8 THEN EXIT key
740 END REPeat key: END DEFine
750 DEFine PROCedure yeano
760 REPeat ysn
770 c$=INKEY$
780  IF c$=CHR$(27) OR c$=="n" THEN
ok=0:EXIT ysn
790  IF c$=CHR$(10) OR c$=="y" THEN
ok=1:EXIT ysn
800 END REPeat ysn:END DEFine
810 :
820 DEFine PROCedure MANE
830 Do$=" "
840 MAIN
850 END DEFine
860 DEFine PROCedure MAIN
870 BORDER 1,2:BLOCK 458,12,0,0,2: STRIP
2: AT 0,37: INK 7:     PRINT "zip"
880 CLSc: STRIP 0: INK 2: AT 2,5:
PRINT"ZIP ": sw: cr=0: Rx
890 k3: pk
900 IF k=-1 THEN CLSf: INK 5: AT 1,0:
PRINT Do$: Press: PAUSE: Rx: GO TO 950
910 LET Zipfile$=Z$(k+1)
920 filename$=t$&W$&"_"&Z$(k+1)
930 AT 2,3:INK 7:PRINT" ZIP ";:INK
5:PRINT filename$ ;: INK 2: PRINT "
(y/n)"
940 yeano:IF ok THEN Do$=Do$&"
"&filename$:ELSE GO TO 950
950 CLSf:AT 2,3: INK 7: PRINT" ZIP ";:
INK 5: PRINT "another" ;: INK 2: PRINT "
(y/n)"
960 yeano: INK 2: IF ok THEN B1: Rx:
STRIP#2,7: CLSd: rx2: GO TO 890: ELSE GO
TO 970: END IF
970 CLSf: AT 2,3: INK 5: INPUT "ZIP file
name: ";name$
980 IF name$="" THEN GO TO 970
990 CLS#2
1000 DoWhat$=name$&" "&Do$
1010 EXEC_W win1_zip;DoWhat$
1020 CLS#2: BORDER#2,1,7: wx: MAIN: END
DEFine
1030 :
1040 DEFine PROCedure B1: BEEP 100,10:
END DEFine
1050 DEFine PROCedure B2: BEEP 200,20:
END DEFine
1060 DEFine PROCedure B3: BEEP 900,20:
END DEFine
1070 DEFine PROCedure B4: BEEP 900,40:
END DEFine
1080 DEFine PROCedure B5: B3: PAUSE 5:
B4: END DEFine
1090 DEFine PROCedure lne:
BLOCK#2,458,1,24,2,0:
BLOCK 458,1,0,10,0: END DEFine
1100 DEFine PROCedure nd: PAPER 7: lne:
B2: tre=1: WCh: END DEFine
1110 DEFine PROCedure CL: BLOCK
450,220,0,11,7: PAPER 7: INK 2:   AT
0,64: PRINT CHR$(188)&' shift TAB ': AT
0,0:  PRINT" TAB "&CHR$(189): INK 0:AT
0,7:  PRINT " SELECT_DEVICE  ZIP "TO
57;" EXIT ": PAPER 2: INK 7:END DEFine
1120 DEFine PROCedure K4: PAPER 7:
REPeat key
1130 ike=CODE(INKEY$)
1140 IF ike=9 OR ike=32 OR ike=252 OR
ike=253 OR ike>47 AND ike<58 THEN B3:
EXIT key
1150 IF ike>=58 AND ike<252 THEN B4: K4
1160 END REPeat key: END DEFine
```

```
1170 DEFine PROCedure Pick
1180 IF ike=253 THEN B4: PrvW
1190 IF ike=9 THEN B3: NxtW
1200 IF ike=252 THEN B4: B5: GO TO 1530
1210 END DEFine
1220 DEFine PROCedure NxtW
1230 tre=tre+1: IF tre=3 THEN tre=0
1240 WCh: END DEFine
1250 DEFine PROCedure PrvW
1260 tre=tre-1: IF tre<0 THEN tre=2
1270 WCh: END DEFine
1280 DEFine PROCedure WCh: lne: CL
1290 IF tre=0 THEN AT 0,7: PRINT"
SELECT_DEVICE ": K4: Pick: cj
1300 IF tre=1 THEN AT 0,22: PRINT" ZIP
": K4: Pick: CLS: MAIN
1310 IF tre=2 THEN AT 0,57: PRINT" EXIT
": K4: Pick: GO TO 1530
1320 END DEFine
1330 DEFine PROCedure cj: pn=7: CLSe:
BLOCK 90,100,42,11,0: BLOCK
88,99,43,11,7
1340 PAPER 0: INK 7: AT 10,7: S$=t$:
PRINT" [ ";: INK 5:   PRINT S$&W$;: INK
7: PRINT"_mode ] ": AT 11,7: PAPER 2:
PRINT u$: PAPER 7
1350 INK 0: BEEP 100,29: AT 2,9:
PRINT'"W" = win1_': AT 4,9: PRINT'"F" =
flp1_': AT 6,9: PRINT'"R" = ram1_': AT
8,9: PRINT'"O" = other'
1360 s=CODE(INKEY$(-1))
1370 IF s=9 THEN B3: BLOCK
90,112,42,11,7: tre=1: WCh
1380 IF s=253 THEN B4: BLOCK
90,112,42,11,7: tre=3: WCh
1390 IF s=77 OR s=109 THEN t$="mdv":
W$="1" : GO TO 1480
1400 IF s=78 OR s=110 THEN t$="mdv":
W$="2" : GO TO 1480
1410 IF s=70 OR s=102 THEN t$="flp":
W$="1" : GO TO 1480
1420 IF s=71 OR s=103 THEN t$="flp":
W$="2" : GO TO 1480
1430 IF s=82 OR s=114 THEN t$="ram":
W$="1" : GO TO 1480
1440 IF s=84 OR s=116 THEN t$="ram":
W$="2" : GO TO 1480
1450 IF s=87 OR s=119 THEN t$="win":
W$="1" : GO TO 1480
1460 IF s=88 OR s=120 THEN t$="win":
W$="2" : GO TO 1480
1470 IF s=79 OR s=111 THEN WINDOW#0,90,
30,69,84: PAPER#0,7:INK#0,2: other
1480 sx: B5: MANE: END DEFine
1490 DEFine PROCedure other:INK#0,0:
AT#0,0,7: PRINT#0,"    "&CHR$(188)&"_";:
INK#0,2: AT#0,0,8: INPUT#0,t$;: IF
LEN(t$)<>3 THEN GO TO 1480: END IF
1500 INK#0,5: AT#0,0,11: PRINT#0,"_":
INK#0,2: AT#0,0,11:
INPUT#0,W$: IF CODE(W$)>56 OR
CODE(W$)<49 THEN GO TO 1510:
WINDOW#0,413,10,50,241:END IF
1510 END DEFine
1520 CLS#2: PAPER#2,7: INK#2,2: AT#2,24,
32: PRINT#2,"@ PLATYPUS Software"
```

QLAMBer Update

by Al Feng

QLAMBer has recently been updated to reflect known changes in QDOS compatible hardware.

What's New

To facilitate greater user convenience, the new RomDisq is acknowledged as a quick_key device in the "other" option (NB: it is my understanding that "rom" is the device name for the RomDisq, and, this will be corrected if I am advised it is different).

```
FLP device supported with 'f'
MDV device supported with 'm'
NDK device supported with 'n'
RAM device supported with 'r'
ROM device supported with 'o'
WIN device supported with 'w'
```

Further, the "CONTROL SHIFT" quick_key combination pioneered in ZipaDee_bas has been incorporated into the current version of QLAMBer (2.205).

flp1_ device accessible with 'CTRL SHIFT f'
mdv1_ device accessible with 'CTRL SHIFT m'
ndk1_ device accessible with 'CTRL SHIFT n'
ram1_ device accessible with 'CTRL SHIFT r'
rom1_ device accessible with 'CTRL SHIFT o'
win1_ device accessible with 'CTRL SHIFT w'

Pressing one of these combinations will "return" you to the EXEC_W screen for the device selected.

QLAMBer is compatible with SMSQ (QXL), Minerva, JS/JSU, QLAY (some limitations exist due to the emulation). If your current version is compatible with what you are currently using and the ROM code is different than those listed, then QLAMBer should continue to be compatible.

How To Get Your Copy

If you have e-mail then you can get a FREE upgrade if you have previously purchased a copy of QLAMBer, QLUSTer, or QLUTter from either EMSoft, UPDATE!, or PLATYPUS Software by simply e-mailing me

alfeng@juno.com

You will receive a ZIPped file that has been encoded for transmission.

You will need to have the facility to decode the file. A UUEncoded file will be sent unless you indicate otherwise (i.e., MIME or BinHex).

To receive an upgrade by disk, please send $2.00 and indicate disk size to: Al Feng
914 Rio Vista Circle SW
Albuquerque, NM 87105

QLATter 1.109

by Al Feng

QLATter is a freeware utility intended for use with Jan Venema's QLAY emulator; but, it can be used with a "regular" QL or any other QDOS compatible, too.
The source code for the program has been corrected and updated so that the COPY function works, and the FORMAT@ function has been replaced with a M@K_DIR (faux/mock sub-DIRectory) option.

At the present time, Hard-COPY remains non-functional due to limitations in the QLAY emulation; but, it will work on other QDOS compatibles.
QLATter supports easy sub-DIRectory access and is Minerva and SMSQ compatible.
TK2_EXTensions are not required.

SELECT_DEVICE '0'

While QLATter's 'SELECT_DEVICE' option does not have "mdv()_" as a ready option, microdrives users can access the two devices on their QL via 'other'.
If you select 'other', then you can simply press 'm' and then the <ENTER> key, followed by either '1' or '2' and then the <ENTER> key, again.

Otherwise, simply move the green bar up or down using either the up_arrow or down_arrow key, or by pressing the first letter of the device name.
Change the device number by using the left_arrow or right_arrow key or by pressing a numeric key whose value is between '1' & '8'.

M@K_DIR [F4]

'M@K_DIR' allows you to create a fake/mock sub-directory name ("fake_name ->) which will then allow you to look at appropriately prefixed files (i.e., with the same "name") as if they were in a MAKE_DIR created sub-directory.

If you have selected the wrong device, then input MORE than ten 10) characters in the name to reset or simply press the (esc)ape key to exit.

The 'M@K_DIR' facility traps for duplicate filenames on the same medium.

Getting A Copy Of QLATter

QLATter is really free if you send an e-mail message to me at: alfeng@juno.com
I will send you a UUENCODED ZIP file which you must be capable of UUDECODing and UNZIPping at your end.
You will also receive a QLATter_txt file.

If you do not have e-mail, then please send $1.00 in the US or four (4) IRCs elsewhere to cover the cost of the disk and postage. Please specify disk size.
You can contact me at:

Al Feng
914 Rio Vista Circle SW
Albuquerque, NM 87105

Happy Trails,
And Computing, To You ...

How to Hack on The ZX Spectrum

Les Cottrell

PART 4 - DECRYPTERS

By this point, you should be familiar with headerless loaders, which take up the bulk of most protection systems these days. However, there is one other important aspect of protection that you need to know about if you are to crack the more complex protection systems. It's encryption.

Encryption works like a secret code. You start off with something unintelligible, then you use the "secret rules" to change it into something that makes sense. So, IBDLFST doesn't make much sense, but if you take the previous letter in the alphabet each time, you get HACKERS, which makes perfect sense. Encryption works in roughly the same way. We've already seen that a loading system occupies a small area of memory. An encrypted loading system will appear as a block of code which makes absolutely no sense whatsoever. There will also be a short program which changes all this nonsense into workable code so the loading system can be run. Some really tough loading systems, have more than one decrypter, such as the Alkatrazz loading system which has 250 of the damn things; in practice I've got through about 25 before going mad and hacking something else (don't worry, there's another way of hacking them which I'll tell you about later.)

To make it easier for you to understand decryption, its best to have a look at a real loading system. As an example, I've chosen Impossaball, which was on the YS#75 covertape. Load up STK at address 50000 (it's a safe address), and see what the BASIC has to say for itself by BLOADing it.....

```
10 CLEAR 64530: LOAD "" CODE:RANDOMIZE USR
64531
20 LOAD "" CODE 16384
21 FOR i=1 TO 40 STEP - RANDOMIZE USR 50000
```

Fair enough - so enter CLEAR 64530:LOAD "" CODE and start the tape. The first code block loads in, and then - it crashes! What's going on? Don't worry, you have just fallen victim to the first type of encryption - BASIC encryption.

In BASIC, any numerical constant (techno-twaddle for "number") between 0 and 65,535 is stored in the memory as follows. First, there is the number in it's ASCII form, which takes between 1 and 5 bytes. So, the number 1234 will appear as #30,#31#,#32,#33 here. Then, come the numbers #0D,#00 and 00 (this is always the case). Then, there is the number stored in it's two byte form (as in machine code). What happens it that the computer prints the ASCII form (which is what you get when you LIST a program), but uses the two-byte form in calculations and expressions. The upshot of all this is that what you see isn't always what you get! As an example, type in this:

```
1 PRINT 1
```

Now type POKE 23760,50 (which changes the ASCII value of the number 1 from "1" to "2"). Now if you LIST the program, you will see 1 PRINT 2, but if you RUN the program, the computer will print the number 1 instead!

Needless to say, protection systems use the same idea. Sometimes, the numbers listed are obviously fake (if you list a program and you get something like 0 CLEAR 0:RANDOMIZE USR 0 it's obviously got encrypted BASIC), but some programs, like the Impossaball are not obvious at all until you try executing what you see.

There was a program called *List printed ages ago in YS, but if you haven't got that, I've reprinted it here. So type it in, SAVE it to tape, RUN it and reload the Impossaball BASIC loader.

loading LINE 10 LEN 81

```
   10 CLEAR 25599: LOAD "" CODE:RANDOMIZE
   USR 64512
   20 LOAD "" CODE 16384
   21 FOR i=1 TO 40 STEP ????....
```

Now that's what you really get! Type CLEAR 25599:LOAD "" CODE and load in the first block of code. When that's done, load up your disassembler, and disassemble address 64512 (FC00 hex) to have a look at the decrypter.

```
   FC00 01 99 02 LD BC,#0299
   FC03 21 13 FC LD HL,#FC13
   FC06 11 14 FC LD DE,#FC14
```

This part of the program sets up all the initial values for the Decrypters in some of the registers.

```
   FC09 1A   LD A,(DE)
```

We've come across brackets before, but briefly what happens here is that the contents of the address with the value of the DE register (which starts of as #FC14) are put into the A register. So now the A register could contain any byte.

```
   FC0A AE   XOR (HL)
```

We haven't seen XOR before, so I'll explain what it does. It is in technical terms a Boolean Operation. You may have seen XOR gates if you studied (or are studying!) Physics, Electronics or Computer Science at school. An XOR gate has two inputs, which can each either be 0 or 1, and one output, which can be either 0 or 1 as well. If the two inputs are the same (0 and 0, or 1 and 1), the output is 0, otherwise it is 1. In machine code, you XOR the A register with a number or contents of a register, and what happens is that each bit in the A register is XORed with the same bit in the number or register contents, and the result is stored in the A register. If you're confused, look at the example below.

Contents of the A register	00100101
Number to be XORed with	01010011
Result	01110110

(Notice that all the numbers are in binary - see your Spectrum manual for more information about this).

If you still don't understand, just remember that an XOR will change the contents of the A register. That's all you need to remember for now.

Continuing the disassembly...

```
   FC0B 77  LD (HL),A
```

This puts the value of A (which has just been changed) into the bytes at the address with the value of the HL register (which starts off as #FC13).

So, the routine has basically taken a byte out of a memory location, changed it a bit, and put the altered value back again. This is decryption in its most obvious form - the changed values make up a working machine code program.

```
   FC0C 23   INC HL
   FC0D 13   INC DE
   FC0E 0B   DEC BC
```

I don't think I've mentioned INC before, but it's basically the opposite of DEC in that it increases the value in whatever register. So the values in the HL and DE registers are incremented, and the value in the BC register is decremented.

```
   FC0F 78   LD A,B
   FC10 B1   OR C
   FC11 20 F6    JR NZ,#FC09
```

This is a standard piece of code, and it essentially means "If BC isn't 0, then jump to #FC09". In other words, another byte will be decrypted until the value of BC is 0 (which happens when everything has been decrypted), and the decrypter ends.

Continuing the disassembly...

```
     FC13 5C LD E,H
     FC14 9F SBC A,A
     FC15 A5 AND L
     FC16 5B LD E,E
     FC17 13 INC DE
     FC18 5B LD E,E
```

Hang on - this code doesn't make sense! It has no relation to the code above, and the instruction LD E,E is pointless anyway. Well, you'll remember that the initial value of decryption is #FC13, which means that all the code from there onwards has to be decrypted. So we'll have to crack the decrypter to go any further.

Sometimes, it is possible to put an EI/RET instruction directly after the decrypter, but this is not possible here, as you will see. So instead, we'll have to move the decrypter somewhere else in memory, and put the EI/RET on the end (in actual fact we don't need the EI because there is no DI command in the decrypter). This is easily done by using the LDIR command. Type in the following program:

```
   1 FOR N=23296 TO 23310:INPUT A:POKE
   N,A:NEXT N
```

Now RUN it and enter the following numbers:

```
33,0,252,17,128,91,1,1,18,0,237,176,54,201
,201
```

The program you have just typed in is this:

```
     LD HL,FC00
     LD DE,5B80
     LD BC,0012
     LDIR
     LD (HL),C9
     RET
```

Now RANDOMIZE USR 23296 and the decrypter will be copied to 5B80 and a RET will be stuck on the end. Just RANDOMIZE USR 23424 (5B80 in decimal) to run the decrypter. When the OK message comes up, restart the disassembler and look at FC13 again:

```
   FC13 C3 3A FE JP #FE3A
```

This jumps to #FE3A, to start the loading.

```
   FE3A F3   DI
   FE3B 21 00 58 LD HL,#5800
   FE3E 11 01 58 LD DE,#5801
   FE41 01 FF 02 LD BC,#02FF
   FE44 36 00          LDIR
```

As you already know, this makes the screen black.

```
FE48 CD 81 FE CALL #FE81
FE4B CD 00 80 CALL #8000
```
If you look at the code at #FE81, you'll see it's a headerless loader. The routine at #8000 prints the loading screen.
```
FE4E CD 81 FE CALL #FE81
FE51 CD 00 64 CALL #6400
```
This loads another block (the main game), and prints another screen (the border for the game).
```
FE54 F3       DI
FE55 31 FF FF LD SP,#FFFF
```
This disables interrupts and changes the stack pointer, so we'll have to change that in the final hack.
```
FE58 21 00 6D LD HL,#6000
FE5B 11 00 5B LD DE,#5B00
FE5E 01 00 8F LD BC,#8F00
FE61 ED BO    LDIR
```
This is another LDIR command, but if you know how the Spectrum's memory is organised, you will see that the BASIC system variables are overwritten, which means we can't return to BASIC when the game has loaded. Fortunately, there's a short routine to get round this, which I'll explain in a mo.
```
FE63 21 71 FE LD HL,#FE71
FE66 11 00 F0 LD DE,#F000
FE69 01 10 00 LD BC,#0010
FE6C ED B0       LDIR
FE6E C3 00 F0 JP F000
```
This moves the code from FE71 to F000, and jumps to F000. Obviously, then, the code at FE71 is important.
```
FE71 21 00 FC LD HL,#FC00
FE74 11 01 FC LD DE,#FC01
FE77 01 FF 01 LD BC,#01FF
FE7A 36 00       LD (HL),0
FE7C ED B0       LDIR
FE7E C3 00 80 JP #8000
```
This routine wipes out all the memory from #FC00 to #FC01, but in actual fact you don't need to do this. Then it jumps to #8000, which is the start of the game. So, we can put a machine code routine to put a POKE in, and then jump to #8000.

However, we've got to find the POKE first, and there's the problem that we can't use BASIC because it is overwritten. Luckily, there is a short routine you can use which will cause a NEW to a certain address. Put it into the above program by typing in the following:
```
1 FOR N=65137 TO 65144:INPUT A:POKE
N,A:NEXT N
```
Then RUN it, and enter these numbers in turn:
```
243,175,17,0,95,195,203,17
```
The program you've just typed in is the following:
```
DI
XOR A
LD DE,#5F00
JP #11CB
```
DI disables interrupts, XOR A loads A with 0 (think about what would happen if you XORed the value of the A register with itself, LD DE,#5F00 means we want to NEW up to #5F00 (this value can be changed from about #5D00 to #FFFF), and JP #11CB starts the NEW.

Now RANDOMIZE USR 65082 (#FE3A in decimal), and restart the game tape. When the computer resets, you can load STK into address 24320 to find POKEs.

Phew! And that's about all of that protection system cleared up. If you can get your way through that lot, I think you're probably ready to have a go at a "commercial" protection system. First, though, we'll write a complete hack for the game.
```
10 CLEAR 25599:LOAD "" CODE
```
This is from the BASIC loader, and it loads in the first machine code block
```
20 FOR N=23296 TO 23310:READ A:POKE
N,A:NEXT N
```
This line POKEs in the machine code program to move the decrypter.
```
30 DATA 33,0,252,17,128,91,1,19,0,237,
176,54,201,201
```
And here's the actual machine code itself
```
40 RANDOMIZE USR 23296
```
This line calls the decrypter and returns to BASIC
```
40 FOR N=65137 TO 65143: READ A:POKE
N,A:NEXT N
```
This line POKEs in our hacking program
```
50 DATA 175,50,??,??,195,0,128
```
And here's the hacking program, which loads ???? with 0 (which is the infinite lives POKE), and jumps to #8000.
```
60 RANDOMIZE USR 65082
```
This starts the whole loading system off, with the POKE firmly in place.

And that's about it. A bit of a long piece of work, but it was worth it!

PART 5 - **Advanced Hacking Methods**

Remember in Part 2, when I said there were other ways of hacking, apart from forwards and backwards tracing? Well, you can find them here. But you need a Multiface to be able to use them. The two we're interested with here are what I call a stack trace and an interrupt trace.

STACK TRACE

We've already come across the stack as a means of storing numbers. What you haven't come across yet is how the stack is used. Well, in a CALL to a subroutine, what actually happens is that the return address from the subroutine (which is the address after the CALL instruction) is stored on the stack, and with a RET, the top value of the stack is taken off and jumped to.

With a Multiface, the value on the top of the stack is the return address to the program. and subsequent values refer to return addresses in subroutines.

To do a stack trace, load and play the game, and wait until the "death effect" occurs - this may be a beep, a flashing border or something else recognizable. Now quickly press the Multiface button during this effect - if you're too slow, you won't get the values you're looking for (so return to the game and die again). Now, look at the value of the stack pointer (your Multiface manual will tell you how to do this), and write down all the values on top of the stack for the first ten bytes. All numbers are stored in the normal reversed two-byte form, so if the bytes on the top of the stack were #00,#80,#80,#70,#90,#60, the values would be #8000,#7080 and #6090. Have a look at all of these addresses - you should find that some of them are addresses right after CALL instructions.

Now for the hacking bit - go to one of these address and write down the two bytes there. Then change them to the magic codes #18 and #FE (this is the machine code version of JR -2, which is an endless loop, a bit like 1 GOTO 1 in BASIC). Restart the game, and hopefully, you'll find that the game pauses as soon as you do something which would normally result in you losing a life! (If not, replace the #18FE with the original two bytes, look at another address on the "hit list" and repeat the whole procedure).

Once you've found a target address, try putting a RET (#C9) at the start of the subroutine. If this just cancels the death effect, but you still "die", activate the Genie Disassembler if you have it (or use the NEW routine in Part 4 at any address, then load in STK or Devpac somewhere far away from the area of memory you're looking at), and search for CALLs to this routine. Then go back from this CALL until you find a RET or a JP, and search for the address of the instruction after this (if nothing comes up, search for one more than this, then two more etc.). You will hopefully either see one of these:

```
JP  Z
JP  NZ
JP  C
JP  NC
```

(The JP may be a JR or a CALL instead)

Simply overwrite this instruction with NOPs to get immunity or something similar.

On the other hand, when searching from the CALL address, you may find a JP Z or JP NZ, etc. Change this to an unconditional JP to get immunity.

INTERRUPT TRACE

This involves looking at the interrupt routine in the game. Since the whole routine must be executed in 1/50th of a second, the routines are usually quite short, especially if there is a LDIR or something similar. Most of the time you'll find infinite time in this routine (because interrupts work in real time, so its an ideal place to put a time routine), and you need a Multiface to find it.

Load the game and start playing as normal. Then activate the Multiface, and have a look at the I register. If the value is #3F, there are no special interrupts, so forget about an interrupt trace altogether (but you can use a stack trace which will make the clock loop round to 99 or whatever when it reaches 0). If it is between #80 and #FF (and if it's not in that range and not #3F you've probably crashed the computer!), go to the address #100 time that of the value in the I register (so if the value of I is #F0, look at #F000). You will see an area of memory filled with the same number. Go to this address (if this area of memory is filled with #FE, go to #FEFE etc.) There will either be a jump to the interrupts routine, or the interrupts routine itself. Have a look at the routine, and somewhere you will see the commands to decrease the timer - just remove the DEC instruction to get infinite time.

RMG Inventory

5 TS 1016 16K RAM Pak (1000)
2 Magic Bridge-16K RAM Packs
4 MODEM-Byte-Back Spectraterm V1.3 (2068)
4 E-Z Key-Upload 2000 (2068)
4 Timex-Circuit Board Scramble (2068)
5 Timex-Crazybugs (2068-T)
1 Wilcox-Tarot (2068)
1 Wilcox-Yacht (2068, S/W)
14 Keytops-Cricket House (2068, H/W)
10 Keyboard-E Z Key Interface Kit (1000, 2068)
1 Printer-TS2040 Complete (Used)
94 Timex-Spelling I (2068-T, S/W)
8 Timex-Crazybugs (2068-T, S/W)
7 Timex-States & Capitals (2068-T, S/W)
6 Book-Sams Beginner/Intermediate Guide
6 Book-Sams Intermediate/Advanced Guide
10 Book-Computer Interfacing In Science (1000/2068)
14 Book-T/S 2068 Basics & Beyond (S/W, 2068)
6 Computer-TS 1000 (H/W)
4 Keytops-Cricket House (1000) (H/W)
1 Compusa-Keyboard Bleeper Kit (1000, H/W)
1 Keyboard-Ti Surplus (1000, 1500, H/W)
13 Book-The Ins & Outs Of TS1000 (S/W, 1000(1500)
20 Book-TS 1000 Owners Manual (S/W)
46 Mindware-Gulp (1000, S/W)
2 Pad-TS 1000 Computer/RAM Desk Pad (H/W)
52 Softsync-Alien Invasion (1000, S/W)
52 Softsync-Red Alert! (1000, S/W)
49 Softsync-2K Games (1000, 1500, S/W)
7 Timex-Budgeter (1000, S/W)
24 Timex-The Carpooler (1000, S/W)
12 Timex-The Challenger I (1000, S/W)
27 Timex-Chess (1000, S/W)
16 Timex-College Cost Analyzer (1000, S/W)
1 Timex-Gambler (1000, S/W)
25 Timex-Grimms Fairy Trails (1000, S/W)
6 Timex-Heating System Analyzer (1000, S/W)
11 Timex-Ira Planner (1000, S/W)
32 Timex-Loan Mortgage Amortizer (1000, S/W)
1 Timex-Mixed Game Bag I (1000, S/W)
16 Timex-Money Analyzer I (1000, S/W)
2 Timex-Money Analyzer 2 (1000, S/W)
1 Timex-Power Pak 1 (1000, 1500, S/W)
30 Timex-States & Capitals (1000, S/W)
28 Timex-Super Math (1000, S/W)

Jack Boatwright
87325 Fryrear Rd.
Bend, OR 97701
541-389-7353 (H)
jboatno4@outlawnet.com

For the inventory at J. Shepard, Iowa
Check with Frank Davis at FWD Computing.